Supplement to
NASA Conference Publication 2147

Research in Nonlinear Structural and Solid Mechanics

Keynote address and panel discussion
presented at a symposium
held at Washington, D.C.,
October 6-8, 1980

NASA

Supplement to
NASA Conference Publication 2147

Research in Nonlinear Structural and Solid Mechanics

Compiled by
Harvey G. McComb, Jr.
NASA Langley Research Center
Ahmed K. Noor
The George Washington University
Joint Institute for Advancement of Flight Sciences
NASA Langley Research Center

Keynote address and panel discussion
presented at a symposium sponsored by
NASA Langley Research Center, Hampton,
Virginia, and The George Washington University,
Washington, D.C., in cooperation with
the National Science Foundation,
the American Society of Civil Engineers,
and the American Society of Mechanical
Engineers, and held at Washington, D.C.,
October 6-8, 1980

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1981

BLANK PAGE

FOREWORD

A symposium on Computational Methods in Nonlinear Structural and Solid Mechanics was held in Washington, D.C., on October 6-8, 1980. NASA Langley Research Center and The George Washington University sponsored the symposium in cooperation with the National Science Foundation, the American Society of Civil Engineers, and the American Society of Mechanical Engineers. The purpose of the symposium was to provide a forum for communicating recent and projected advances in applied mechanics, numerical analysis, computer hardware and engineering software, and their impact on modeling and solution techniques in nonlinear structural and solid mechanics.

The fields covered by the symposium are rapidly changing and are strongly impacted by current and projected advances in computer hardware. To foster effective development of the technology, therefore, an expert in computer hardware was invited to communicate in a keynote address his perceptions on new computing systems and their impact on computation. Also, five experts in nonlinear analysis software systems were invited to participate in a Panel Discussion on "Future Needs and Directions of Nonlinear Analysis."

This NASA publication includes the keynote address "New Computing Systems and Their Impact on Computations"; introductory remarks presented by the panelists; and selected comments from the audience and the responses by the panelists during the Panel Discussion. Two other symposium proceedings which include 68 of the papers presented at the symposium have already been published. These are:

Noor, Ahmed K., and McComb, Harvey G., Jr. (Editors), Computational Methods in Nonlinear Structural and Solid Mechanics, Pergamon Press, 1980, containing 48 papers; and,

McComb, Harvey G., Jr., and Noor, Ahmed K. (Compilers), Research in Nonlinear Structural and Solid Mechanics, NASA CP-2147, 1980, containing 20 papers.

Any opinions, findings, conclusions, or recommendations expressed in these publications are those of the authors and do not necessarily reflect the views of NASA or The George Washington University. Use of manufacturers' names, trade names, or computer program names does not constitute an official endorsement of such manufacturers, products, or computer programs, either expressed or implied, by NASA or The George Washington University.

Harvey G. McComb, Jr.
Ahmed K. Noor
Compilers

BLANK PAGE

CONTENTS

BLANK PAGE

NEW COMPUTING SYSTEMS AND THEIR IMPACT ON COMPUTATION

David Stevenson

Zilog, Inc.
10460 Bubb Road
Cupertino, California 95014

ABSTRACT

The computer industry can be divided into three strata: supercomputers, mainframes and minicomputers, and microelectronics. Each segment has its own course of development. This paper describes some of the more important developments of each segment in the near future from the point of view of the potential opportunities presented for scientists engaged in numerical experimentation. In each case it is seen that the potentials available in the hardware are limited mainly by the software tools that may be available.

INTRODUCTION

Computer technology can be best described, if at all, by the term "explosive growth." For all practical purposes this means that the "new" has a currency of between two and five years (at the most) and that to talk about the impact of new computing systems beyond five to ten years is already stretching credibility. This paper then, is less ambitious than its title in that it reviews the highlights of developments in the last five to ten years and projects their implication into the next decade using a "no breakthrough/surprises" model of progress, which by the past performance of the industry, will be interesting but short-sighted when viewed from the vantage point of 1990, but will suffice for the present discussion.

The first section develops a model of progress in the computer industry, based upon innovation, technology, and commercial interests. This model can be applied to several of the different strata of computers, from large-scale monster computers to personal computers. This is done in the next three sections which are devoted to large-scale scientific computers, general computers, and microelectronic computers. Each of these areas present tantalizing opportunities for the development of computer structures that are amenable to structure computations; the main problem is seen to be not computer technology (hardware), but software: the user's investment of his knowledge into computer-readable form. Hardware development, due to the rapid pace of technology, has tended to pace software development to the point that it's safe to assert that the "problem" is software rather than hardware.

The paper begins with a brief discussion of how new computer structures arise, both from a technical point of view and from a commercial point of view. Next, three segments of the computer market are surveyed as to new computing

systems and their impact on scientific-oriented numerical computation. These segments are: large-scale scientific computing, general computing (mainframe and minicomputers), and microelectronic based systems.

The Computing Industry

Perhaps no description of the computer industry is more frequently invoked and more aptly used than "explosive growth." A rapidly advancing technology of electronics makes possible the rapid introduction of new products that offer more computing power, better price performance, and a breathtaking drop in the cost of acquiring a minimal computer adequate for performing numerical experiments in structural analysis.

The technology developments have been in the improvements of cost, size, power consumption, speed, and reliability of electrical components. These advances have broadened the options for the designer of computing systems. There are two ways for the designer to proceed: implement a revolutionary new idea or build upon existing, proven concepts to offer an evolutionary improved product. Three of the more recent revolutions (within my professional lifetime) have been timesharing, vector processing, and distributed computing. Each has had a profound influence on the way we perceive "computing" although the influence has not been universally felt. Timesharing required giving up substantial computing power to the operating system in return for immediancy (putting the user in touch with the computer solving his problem). Vector processing required using "less efficient" algorithms (parallel operations, some unnecessary) in return for speed (getting the total problem solved sooner). Distributed computing, the current revolution, allows us to abandon concern for efficiency in return for convenience (physically bringing the computer to the computor).

In order to appreciate where the new computer structures are likely to evolve, it is convenient to have a simple model of the computer manufacturers. The key idea is the notion of a product life cycle, during which a company recovers its investment in developing a system and goes on to make a profitable return on its investment. The more investment required to bring a product to market, the more expensive, or more universal, or longer lived it must be. In order to amortize an initial investment over several products, systems usually come in families (in mature technologies) or as a succession over time of improved versions of the same product (exploiting the rapid advances of a new component technology).

As hardware systems become more complex, the development effort and amount of software required to make the system operational and "useful" increase. This tends to prolong the lifetime of complex products. Thus, the new structures discussed below will likely be around in some form for quite some time.

Large-Scale Computing

The most recent revolution in large-scale computing has been parallel processing: single instructions operate on whole vectors or arrays of numbers.

The earliest such machines in the early 1970's were the ILLIAC IV (a processor consisting of an array of simple units under the lock step control of a centralized processor) and the CDC STAR-100 and TI ASC (pipelined processors where vectors of operands were processed in an assembly-line fashion). The first widely accepted large-scale parallel processor (placing more than ten machines in the field) was the CRAY I in the late 1970's. The early eighties will see the second generation of vector processors come on line: the CYBER 203 (successor to the STAR-100) and the BSP (Burrough's successor to the ILLIAC IV) have already been announced.

The major advantage of vector processors is that they offer substantially more computing power than conventional scalar machines (by a factor of five to ten in commercially available machines). 1980 has machines capable of over 100 million floating-point operations per second, and a rate of one billion floating-point operations per second from a commercially available machine is possible before the end of the decade.

But all of this power is not without its cost: vector processors require that programs be written in such a way as to exploit the potential of the machine. Problems must be formulated to operate on vectors. The first vector machines, such as the STAR-100, required very long vectors in order to operate efficiently. This often led to artificial programming constructs. For example, when sweeping through a 128 by 128 mesh where each row could be processed in parallel, blocks of rows would be grouped into a single long vector for processing. Although this is not difficult, it leads to complex bookkeeping and complicated programs to write, debug, and maintain. Future vector processors are likely to correct this problem, but much algorithm development time has already been spent in developing such tricks.

A particular type of vector processor may well become increasingly more prevalent in the 1980's: this is the "array processor" which is implemented using arrays of simple arithmetic elements to process a vector, each element of the vector being processed by a different arithmetic unit, all units operating in parallel. The advantages of an array are its inherent modularity (group elements together) and, hence, extensibility (keep adding more groups); the disadvantages are its relative inflexibility and apparent difficulty in programming effectively. The disadvantages have to date been so severe as to impede the spread of array processors except in special purpose configurations. Two technology-based factors may overwhelm the disadvantages: an array architecture is the simplest method of achieving very high computing power for a given component technology, and semiconductor technology now permits considerable computing power "on a chip" at a very inexpensive price (c.f., your hand-held calculator whose computing power rivals that of the earliest computers). These two developments present arguments for array processors at the high end and low end of the computing spectrum, respectively.

As to the impact of vector processing on a large-scale computer, a number of things can be said. First, very large problems requiring extensive calculation will most likely be performed on vector processors in the future. As the decade progresses, vector processors will become more widespread and hopefully more hospitable. Vector processors require regularity: algorithms free

of "special cases" are preferred, both to simplify programming and to aid efficient processing. For example, the trade-offs involved between global approximations to a function versus local approximations will have to be re-accessed with each new vector processor. Secondly, old software "originally written for the 7094", will have to be rewritten (rather than patched again to take advantage of the new vector capability). Now would be an ideal time to rethink the problem and the means of solution rather than merely recode the old method to take advantage of the advances in numerical methods and computational physics over the last fifteen years. (Having said this, I will now admit what is obvious: I am not a computing center director administering a large base of user donated software; I understand his problem, but have no sympathy for him.)

Finally, the major outstanding problem with vector processors is the lack of programming languages adequate to the needs of expressing efficient algorithms for vector processors. APL, the classical example of a vector-oriented language, has serious flaws from a structured programming point of view; consider the branch mechanism and imbedded assignment facility, to name two examples. Since the user community for large-scale computers is small compared to mainframe users, one cannot expect rapid advances in software support for these machines. The consequence of this is that software development for vector processors is likely to remain more painful than pleasurable. On the other hand, the majority of scientists have labored under the burden of FORTRAN (resisting ALGOL, for example).

General Computers (Mainframe and Minicomputers)

Large-scale computers have traditionally been the spawning ground for architectural innovations that migrate downwards into the mainframe and the minicomputer. Thus, vector processing has recently appeared in the general computing market. This has taken two forms. The earliest was as an attached peripheral device (such as Floating Point System's AP-120B). More recently, vector instructions have appeared in the native instructions set of the processor (Data General Eclipse, for example). Everything said above about vector processing applies to this small area of the general computing market as well.

A more interesting development, largely confined to the minicomputer market, is the user writable microcode facility. Microcode is that part of a control unit that determines how a CPU will execute an instruction. Thus, user access to this facility has been most widely exercised by system programmers to implement various operations frequently performed by the operating system. However, the potentials for the computing scientist are enormous. As an example, models of structures frequently lead to sparse matrices, for which there are several different data structure representations. It is risky for a manufacturer to select one representation to support in hardware (the STAR-100 tries this, selecting a bit vector representation, which many researchers rejected as too inefficient). It is unwise, if not impossible, to support all possible representations directly in the hardware. On the other hand, a user with access to writable microcode can define an instruction that will search a linked list (in his format) to find a pivitol element for Gaussian Elimination, for example. Because this search is executed by a single hardware instruction, it can be performed much faster than its software equivalent.

User specifiable microcode offers great potentials for the researcher to experiment with new data structures and composite instructions (e.g., SEARCH, SORT, and FFT). But microcode is very difficult to write because of timing and interaction considerations and, hence, is likely to remain the domain of a select population of system programmers. However, knowing such facilities exist may serve to generate interest among the scientific community to experiment with the opportunities that arise here. User accessible microcode is available on DEC's VAX-II/780 and Data General's ECLIPSE.

Microelectronics

Microelectronics is essentially the ability to cram a lot of computing power into a very small piece of silicon circuitry and sell the packaged device for less than the price of a pair of shoes. The impact of microelectronics will be staggering during the next decade, but the exact shape, or even outline, of microelectronics-based scientific computing in 1990 cannot be discerned. In the very near future certain events will become possible. These will be discussed along with some of their possible consequences.

The primary impact of microelectronics stems from the low cost required to construct a reasonable computing system. Unfortunately, the scientific-oriented market is not a mass market, thus it will have to ride the coattails of developments aimed at larger markets which can underwrite development costs. Fortunately, microprocessors are rapidly acquiring characteristics traditionally associated with current minicomputers and mainframes, such as large address spaces, operating system support, and floating-point hardware capability. This means that sophisticated computing power could soon become available at a cost reasonable to dedicate one (or several) full computer system to each researcher. Furthermore, the state-of-the-art in networking is rapidly advancing so that these dedicated computers need not operate in isolation but could share access to common databases.

But a number of problems remain to be solved. First, the issues of privacy and database integrity may not be resolved as fast as networking practice spreads, or to the degree of perfection that users implicitly expect, leading to failed expectations and underrealized potential. Secondly, large sophisticated packages such as NASTRAN (or a "micro-NASTRAN") need to be developed for microprocessors; the proliferation of numerous types of micro-CPU's does not help the situation. Finally, companies must take a profit. It will be interesting to see what type of company will service the small scientific community rather than applying its resources to large markets.

I would like to end by briefly discussing three encouraging developments: semiconductor memories, graphics, and floating-point arithmetic. The cost per bit of memory continues to decline by roughly a factor of two every two years or so. This means that larger and more complex models can be solved as computers of the same price range continue to expand their memory capability.

Graphic terminals and the techniques for displaying data via graphics are undergoing rapid development at the current time. This suggests that visual

display of the results of structural computations can become more frequent as this art advances. The benefits to the researcher to be able to view the results of his computational experiments graphically could become more important than progress in numerical methods in the area of structures.

Finally, a group of computer scientists at the University of California at Berkeley have recently formulated a sophisticated proposal for a floating-point system of arithmetic, including such ideas as a graceful handling of underflow, infinities, accurate rounding, and symbols which permit an escape to user-defined routines (to implement complex numbers, rational arithmetic, or interval arithmetic, for example). Several semiconductor manufacturers have implemented various features of this proposal; the format, rounding accuracy, and encodings for special symbols being the common denominator for these implementations. The existence of inexpensive floating-point hardware that provides a clean arithmetic facility may work wonders for the general scientific community.

CONCLUSION

The coming decade is likely to see major changes in the computing environment. The spread of vector processing capability, the potential for user writable microcode, and the explosive growth of microelectronics were three major developments discussed in this paper. The hardware technology continues to evolve faster than the software technology needed to support fully the added capabilities. For the computing scientist the next decade promises to be full of exciting potential as the revolutions of the previous years evolve into more mature capabilities.

FUTURE NEEDS AND DIRECTIONS OF NONLINEAR ANALYSIS - PANEL DISCUSSION

Part I - Opening Remarks

Robert E. Nickell:

Approximately 7-1/2 months ago, a distinguished group of some twenty-five individuals met to discuss, among other items, unfilled needs in computational mechanics. During these discussions, the state of nonlinear computational mechanics received a considerable amount of emphasis. The group was heavily academic. The findings were that the unfilled needs could be organized into three areas: (1) research needs, which would essentially have to be met by the university community; (2) training needs, which would be met by a combination of university and industrial sources; and (3) communications needs, which were orphans, claimed by no one.

It seems to me that these needs cannot be completely addressed by a group dominated by academic interests. In particular, the needs as expressed by the marketplace can perhaps be articulated best by a select group of first-party software developers who deal for their livelihood with a broad spectrum of second-party users of software. The four individuals that we have with us today are: Professor Klaus Jurgen Bathe from M.I.T., Dr. Joseph F. Gloudeman from MacNeal-Schwendler Corporation, Dr. John A. Swanson of Swanson Analysis Associates, and Dr. H. David Hibbitt of Hibbitt, Karlsson and Associates. Our first speaker is Professor Klaus Jurgen Bathe from M.I.T.

Klaus Jurgen Bathe:

There are several topics that we can and should discuss when we consider important future directions for research in nonlinear analysis. The specific topic that I would like to address is the importance of thorough and effective implementations of nonlinear analysis procedures in university-based finite element research. I believe that this is a very important topic and that, unfortunately, it has not received enough attention during the last decade.

When we think of research in nonlinear analysis, we are thinking of performing research in three different categories: approximation theory, numerical methods, and computer techniques. In Figure 1 I summarized various subcategories of these major categories. In approximation theory, for example, we are thinking of formulation of nonlinear continuum mechanics equations of motion, development of finite element equations, and identification and development of material models. In the category of numerical methods, we are thinking of research in the numerical integration in space, time integration methods, methods for solution of equations, and methods for the calculation of eigensystems. Finally, in the area of computer techniques, we are thinking of programming methods, usage of available hardware and software, efficient program organization and, of course, program development that allows for flexibility for modifications.

I believe that we'll have to perform very significant research in each of these categories during the next decade and thereafter in order to continuously advance the state of nonlinear analysis procedures. For example, in the area of the development of material models, we certainly need more predictive and effective material models for soils and rocks, plastics, composite materials, and so on. In the area of solution methods for nonlinear equations, we certainly want more effective, cost-efficient, and reliable methods for the solution of the equations that we encounter in various areas of nonlinear analysis, and so on. However, when we do perform research in one of these categories, we should also recognize that there is a strong interaction between that research and the research in other areas. For example, we might have developed the theory of a new material model and when we implement that material model we may find that its use would be extremely costly. Therefore, we need to go back to the theory of the material model and rework the theory, then go back to the implementation, rework the implementation and, thus, we have an interaction or iteration between the research in which we develop the theory of the new material model and the computer program implementation of that material model. It is this iteration between the development of a new theory and the implementation of that theory in a computer program which I consider most important.

Another important point, I believe, is that to advance the state of nonlinear analysis procedures we need emphasis on basic, general, and reliable methods. In Figure 2 I have underlined "reliable" twice because I believe that the reliability of the methods that we are proposing and developing is of utmost concern (Refs. 1, 2). This applies to the development of continuum mechanics and finite element formulations, the development of constitutive models, the development of numerical methods, and finally, the computer program implementations. For example, in the area of constitutive models we certainly want general nonlinear models that are applicable to general stress and strain paths and that can, therefore, be used in a reliable manner in practical analysis. The computer program implementations are important steps, and these are followed by the verification and qualification of the models.

It is most important to recognize that the verification and qualification of a theory and the computer program implementation can be a major research task. I believe that generally too little emphasis has been placed at universities on the verification and qualification of analysis methods. However, it must be recognized that this is part of the basic research to be performed in order to advance the state of nonlinear analysis procedures. The insight gained into the nonlinear analysis procedures through the implementation, verification, and qualification of the methods is most important. I believe that it is the implementation, verification, and qualification that assess the applicability and the reliability of the methods. Let me give a few examples.

In the first example I would like to mention the importance of equilibrium iterations in nonlinear dynamic analysis. In 1972 we started the development of the computer program NONSAP for nonlinear static and dynamic analysis. A number of papers had already appeared on nonlinear dynamic analysis, and in each of these the authors had not used equilibrium iterations. However, when we actually implemented nonlinear dynamic analysis procedures, I very quickly

recognized that it is of utmost importance in certain dynamic problems to use iteration methods and, therefore, we did implement in the computer program NONSAP the equilibrium iteration routine which then indeed turned out to be a very important aspect in the computer program. Of course, basically the same iteration scheme is still available today in the ADINA program. The computer program NONSAP was published in 1973 and 1974, and in the years thereafter a number of papers and investigations appeared in which the authors further studied the importance of equilibrium iterations in dynamic analysis, analyzed the schemes that are used in the dynamic time integrations, and so on. This is a striking example in which the implementation of nonlinear analysis procedures really showed the importance of certain strategies, and once the importance of these strategies was recognized, theory could be developed and the procedures could be further enhanced.

Another area where the implementation is of utmost importance is the area of nonlinear constitutive developments. If we think of a complex nonlinear constitutive relation, it is very important to implement the relation and to perform experiments with that constitutive description on the computer in order to gain insight into its predictive capabilities and the cost of using the model in general nonlinear analysis.

As another example, we might refer to methods for solution of eigenproblems, or the solution of nonlinear equations. If we believe that we have an effective strategy for the solution of eigenproblems or nonlinear equations, it is most important to actually implement that strategy and study its performance characteristics.

Considering the verification and qualification of nonlinear analysis procedures, we should be aware that we are really talking about the verification and qualification of a theory, that is, the theoretical assumptions, and the verification and qualification of a computer program. The actual development of a nonlinear solution strategy starts with the development of a theory, and then a computer program is developed. For example, we might develop the theory of a new inelastic material model and then implement that theory in a computer program. We then face the task of first verifying the computer program, and only after we have made sure that the computer program really operates on the theory that we want it to operate on, can we verify and qualify the new theory.

The way a computer program is usually verified is by making a large number of runs. However, this is certainly not sufficient. In particular, we should remember, as summarized in Figure 3, that inappropriate theoretical assumptions plus program bugs may give a good correlation with experimental results. In other words, developing some theory, quickly implementing it, using the resulting computer program in the analysis of a problem for which experimental results are available, and obtaining good correlation with the experimental results do not prove "that our theory and our computer program are already reliable tools."

In the development of nonlinear analysis procedures, we should really distinguish between developing a procedure for analysis of a very specific problem and designing a method that is generally applicable to a whole problem area. Frequently, when a procedure is developed for the analysis of a very specific

problem, we find that the research or program developer and user are really the same person (see top of Fig. 4). On the other hand, when we are thinking of the development of nonlinear analysis techniques that are applicable to a whole problem area, which can be a much more difficult task, we are really thinking of a researcher who develops these techniques and the user who "just" employs these techniques (see bottom of Fig. 4). In this case the researcher has to give high priority to considerations of accuracy, stability, and reliability because the user will very likely not be very familiar with the methods, and it is the researcher who must ensure that the methods are generally applicable. In the future I believe that more emphasis should be placed on the development of these kinds of nonlinear solution strategies.

In summary, to advance the state of the art I believe that we need basic and general theoretical developments, error free implementations, and the experience in the usage of the techniques. I believe that the implementations and the serious usage of the procedures lead to important insights into the methods and, therefore, help in the developments. This insight into the methods, is certainly most important in the development of nonlinear analysis procedures.

Let me now close with some final thoughts. Considering the research performed at universities in finite element methods, I mentioned the importance of actually implementing the theoretical procedures proposed (this was the important message I wanted to present to you). However, concerning the finite element work at universities, or work with finite element methods at universities, it is also important to have the ability of simply using programs. This can be important in general research studies, for example, in the studies of fracture phenomena, and in the teaching effort. In a graduate course recently started at M.I.T. entitled "Theory and Practice of Continuum Mechanics," the theory of continuum mechanics and the practice of continuum mechanics through finite element procedures are taught (Ref. 3). In this course, the use of finite element procedures helps us to teach the theory of continuum mechanics in a very physical way. This, I believe, is a very exciting way to teach, in a practical manner, important theoretical concepts.

R. E. Nickell:

Our next speaker is Dr. Joseph F. Gloudeman from MacNeal-Schwendler Corporation. Joe received his Ph.D. from I.S.D. He was general manager of the data management systems group at Rockwell International for approximately twelve years. He has been with MSC for about two years and is now vice-president for Marketing.

Joseph F. Gloudeman:

Just a followup comment on what Professor Bathe had to say. I really like the notion he had of the development and qualification, to the greatest extent possible, of the work that he is doing in his research program. We tend to be a little bit conservative where I come from and don't like to send something out that is going to be used in a general purpose commercial system until a lot of other people have agonized over it to both develop the capability and prove whether or not it works. It is important to stick with the winners in the eyes of the user community. I really think that addresses one of the specific issues that Bob asked me to look at: "What penalties do we pay in the commercial marketplace?" One penalty is a time lag; we also see here a partial answer to the question: "What can we do to minimize our exposure and risk?"

The outline I am going to follow is shown in Figure 5. First, I am going to follow up on what Dr. Stevenson stated in the keynote address: that the computer hardware, the developments in the computer, and the costs associated with them are really very strong forcing functions. Secondly, in looking at the user requirements, we can follow up on what Dr. Corum (Ref. 4) and Dr. Mikulas (Ref. 5) had to say. Thirdly both Dr. Mikulas' and Dr. Noor's (Ref. 6) presentations as well as Dr. Noor's survey paper (Ref. 7) in the symposium proceedings address very nicely the issues that face the systems developers.

I've got a slightly different approach perhaps than Dr. Stevenson in taking a look at the computer marketplace. The first trend is continuing improvements to the CPU - increases in speed of computation and mean time between failures (Fig. 6). With today's technology we are around 10 mega FLOPS, 10 million floating point operations per second. There are some that advertise a little bit more than that and I suggest that you ask them to go out and prove that to you.

Charles Lecht's "Waves of Change" (Ref. 8) takes a look at the future. By 1985, with the aid of such technology as the Josephson tunnel devices (Fig. 7), we should be looking at CPU's operating in the vicinity of about one billion floating-point operations per second. He quotes Fubini, a key researcher for a major computer manufacturer, who speaks of 100 billion FLOPS by 1985 with the cost getting down to $30 per megabit which is fairly reasonable. In the keynote address, Dr. Stevenson stated that by 1990 we'll have one billion FLOPS, so I'd like to see Lecht and Stevenson get together on this subject. What we're seeing is a reduction of about 30 to 40 percent per year in the cost of memory since 1970 so that does make life a little bit more interesting. Many engineers are turning to super minicomputers often as a means of escaping some of the shackles of centralized data processing.

The impact of having larger main memory is a lifting of some of the restrictions that we currently have because our electromechanical peripheral devices don't function quite as fast as we'd like to see them.

In summary of what's happening with the computer hardware, we can see that the CPU speed, the memory size, and the peripheral storage devices are all moving in the right direction but there is no doubt that the software is lagging. Most hardware manufacturers state that their hardware development costs are less than their software costs. Another trend is the increasing use of hardware and firmware as replacements for software in operating systems which makes them much more efficient and reliable.

There are two particular areas that I think need addressing by our technical community (Fig. 8). One is the area of technically oriented database management systems. You've no doubt heard about all kinds of database management systems for such applications as banking and manufacturing. We need an equivalent for engineering especially when we look at nonlinear mechanics and watch the tremendous growth in the volumes of our data. The second is a need for more attention to standards in such key areas as geometry representation and interfaces, graphics devices, and telecommunications.

The next area is the change in user requirements (Fig. 9). Some of the papers presented at this symposium depict what various industries really need. The area of crash dynamics is of importance, both for automobiles and for helicopters. The nuclear power industry continues to work with problems that involve extremes of temperatures, mechanical loadings, and pressures. Dr. Corum's perceptions (Ref. 4) were extremely valuable in this regard. Several presentations (e.g., Refs. 6, 9, 10, 11, and 12) address the need for perfecting our material models for nonlinear analysis and I certainly agree. There are many unknown areas that need more attention.

The whole area of fatigue and fracture mechanics is lacking generally accepted approaches and techniques. There are military concerns about what happens when lasers hit certain targets. Excellent progress has been made in adapting a general purpose code to analyze the crushing of automobile frames (Ref. 13). Similar success has been achieved in analyzing huge earth moving equipment that involves some very peculiar nonlinearity effects (Ref. 14). Offshore platforms are growing more costly and complex especially since recent oil pools have been found under 300 meters of water. That's pretty challenging in such areas as geometric nonlinearity. The ship structures of liquid nitrogen tankers experience dramatic behavior modes when loading at cryogenic temperatures, even with pre-chilling techniques and the best of spray-on foam insulation. Piping analysis is important in such areas as the energy business where pipe-whip is a problem.

Our next subject is the challenges faced by systems developers in coping with the changes discussed above (Fig. 10). Physical behavior modes are fairly well understood. I do believe that more research is needed in some of our techniques and algorithms. We've heard some good discussions on how to get more flexibility, capability, and reliability into material models and then incorporate selected material models under user control into general purpose codes.

This is a nice goal but it also presents other problems in validation, acceptance, and maintenance. Problems that involve time and temperature dependencies over long time periods introduce both physical and numerical instabilities. Coping with the "memory" of the material is also a real challenge, and we need better experimental correlation along the lines that Professor Bathe described. There is also the question of using reduction techniques. For example, in linear analysis it is usually profitable to work with generalized coordinates rather than the physical coordinates. Reduction techniques are also valuable in the nonlinear area to get the problem down to a smaller size and still get reasonable results. Dr. Mikulas' paper (Ref. 5) addresses the need for a better structural efficiency and how we go about design optimization for nonlinear problems. We haven't yet solved this problem completely in linear elastic analysis.

The next topic we'll cover is the challenge facing system developers in increasing the realistic upper limits of problem sizes (Fig. 11). I believe that with the effective use of superelements (substructuring) and the availability of a large enough computer, there is practically no upper limit on problem size in static plasticity. Even on superminicomputers, 3000 degrees of freedom is not out of the question. Harry Armen of Grumman Aerospace Corporation has successfully solved crash dynamics problems of up to 3000 degrees of freedom using nonlinear transient response. The limits for probabilistic analysis, when dealing with a material property that we do not accurately know, are probably about the same as for the nonlinear transient problem.

In summarizing user needs, it appears that there is an equal need for three areas: plasticity alone; geometric nonlinearity alone; and combined solutions involving plasticity and geometric nonlinearity. There are user needs that have to be continually addressed. One of these is economy, because the user can't always afford to pay the price for the complete analysis that may be needed. Other areas include the dependability of the analysis, the usefulness of the output and its readability, and better ways to cope with growing litigation issues that seem to be facing different industries. Of course, the costs of the analysis must be less than the maximum financial exposure of a potential lawsuit.

The question of where to put the emphasis in development requires crisply-stated inputs from the user community (Fig. 12). For example, what should be the balance between investment in new numerical analysis methods versus the expenditures for improving existing algorithms? How can we achieve wider identification and acceptance of the preferred techniques and establish some form of standards? Coping with the database management area and with the standards for geometry calls for fairly broad involvement.

R. E. Nickell:

Our third speaker is Dr. John A. Swanson, president of Swanson Analysis Systems, Inc.

John A. Swanson:

I was assigned the area of adapting general purpose software to a changing technology base. The first question I want to address is: "Why is the technology base changing?" It would be so nice if things would stand still and we could sit, sell manuals, and make money. But that's not the case. I differ a little bit with Joe. I think the major driving forces are the new problems that keep coming out of the marketplace. That has been the whole theme of the conference: new problems, new geometry, new materials, and so on. My second item is, however, new computer hardware as well, and a third reason why the technology base is changing is something that I would call analysts' productivity requirements. We have to get more and more work out of the same number of analysts because there seems to be some shortage of people that are qualified to do even linear analysis to say nothing of the nonlinear analysis requirements. So the technology base clearly is changing. The next question is: "Why should the general purpose software change?" The answer to that is clear - competition. For example, if Joe changes to meet the market requirements and I don't, then Joe is going to get rich and I'm not. On the other hand, part of what you are hearing is a sense of challenge and a sense of excitement. We enjoy meeting these needs and, on the other hand, we can't move so fast that we lose track of reliability. We want to develop and do new things but we have to make them reliable and usable; they have to be documented, tested, and so on.

I think Joe covered new directions clearly - the hardware forcing function: the minicomputers (the VAX/PRIME/HARRIS, Data General, and so on), the array processors (Floating Point Systems, CSPI, and so on), and the supercomputers (CRAY, CYBER, and the macroprocessors). I was just asked how small we can go, and currently we can go 48K 16-bit words and still run finite elements. That's not quite 32K but it's not too bad either. In the future, we are pushing very strongly for what we call matrix machines. We want firm, hardwired, matrix operations as part of the standard hardware library on the machine. Software trapped, if necessary, to emulate it, but I would like to see a matrix board or a vector board in addition to the floating point processors, just two additional boards in the standard machine.

Another direction of change is software and there's no way to cover the entire range. Certainly we are covering more areas of analysis and more nonlinearities: magnetic fields, dynamics, flow rates in oil wells, and so on. Magnetic fields themselves have their own discrete set of nonlinearities along with all the other nonlinearities we've talked about. In the element area we are seeing enhanced elements more from an incremental standpoint, that is, incremental improvements as opposed to massive improvements. By incremental improvement I mean taking a solid element, building in the reinforcing bars and the failure mechanisms, and calling it a special concrete element, or a special composite element, that is, taking existing technology and packaging it a little different to be more useful. The biggest change in the last few years has been the coming of interactive operation, hands-on modeling of

structures, and graphic displays. In the future, we'll see more of the CADCAM interface, the standardized geometry that Joe referred to. Also in the area of interactive operation is the internal documentation, the self-teaching aspects of computer software. The user's manuals which we sell should be purely reference documents. Hopefully, they should gather dust on the shelf. The programs should give us the answers we need without reference to user documentation.

Whole new areas of accuracy assessment are developing. How do we know how good our result is? If we don't know how good it is, we don't know whether or not we should refine our model. We have to be able to assess the accuracy in order to do mesh refinement, the next step in the automatic procedure.

And finally, there is a whole area of interactions which we have to look at. If you run a large deflection dynamic plastic analysis, you get some interactions going on. The convergence criteria are not as clear. Convergence for plasticity and creep is fairly clear, but if creep is throwing more plasticity into part of the structure, some things happen that are not as clear as if you have individual efforts. Those criteria for convergence have to be reevaluated.

Another direction is education. The software suppliers have to provide education. We are looking into video, workshops, manuals, and internal documentation. Industry has to provide education - workshops within the industrial environment and perhaps more video as well. We find a rather sharp dichotomy in the university environment at the present time, and there are the theory-oriented universities which concentrate on the technique, the writing of special programs to illustrate the theory. There is the other application orientation in which the students are trained in existing software. How does it work? What actually are the results, and so on? In support of this line of activity, Joe and I are supplying low-cost software to the universities (dollar a year or donated, for example). The same system software that's available in the commercial market is also available at nominal charge to the universities. Some take this approach; some prefer to work with other approaches.

Increased testing is leading to improved quality of software products. One of the impacts of the minicomputers is simply the quantity of testing that we do. When software developers were buying computer time on the open market, each computer run represented dollars out of our pocket. Now we think nothing of running a hundred test problems every time we make a minor change in the software. We can automatically verify those problems versus the previous version just by a simple compare operation. We can make sure that a change in Section A did not degrade Section B merely by going through an entire Q/A (Quality Assurance) on the entire system every time there is a minor change. This would be unheard of without owning our own computers. Now it's a very practical approach.

We also have to diligently issue the things we call error reports. We would like to believe that there isn't any such thing as an error but unfortunately, there is. We need some communication path, relatively high speed and reliable, for saying: "Hey!, the load in this particular element is overestimated," or whatever the particular problem.

Another major trend that we are seeing is in problem size, complexity, and range. The basic philosophy is: the problem size expands to fill the computer available. Also, we are seeing more of what I call high technology analysis: substructuring, large deflections, large rotations, large strains, fluid structure interactions, forming processes, and so on. These are not just in the nuclear area or aerospace area either. We see these analyses used in everything from toys to salt mines, and from electrical chips to magic mountains. These are all done with standard structural software.

I would break down software suppliers into the following five categories:

1) Long term general purpose

2) Shorter term specialized

3) Development software for theory evaluation

4) Teaching software

5) Special (one-time) software

The software industry is now a long term industry. For example, we will be starting to program Revision 4 of ANSYS in two years, reaching the market in about five years. It will be entirely different as far as internal structures. The external user may find it similar to the current ANSYS. Software products should be revised every five to eight years to incorporate new developments in computer science. So we are talking about a long term industry, the software development industry. It's no longer the so-called cottage industry. We are talking about five-year development plans, ten-year plans, major interaction with the user community to supply us with the input as to what's needed, and major competition among the vendors to try to keep alive the 25 or 50 people on our staffs, and to provide high quality software to the user community.

R. E. Nickell:

Our final speaker is Dr. H. David Hibbitt of Hibbitt, Karlsson and Associates. Dave received his Ph.D. from Brown University in 1970 and worked for the MARC Analysis Corporation a number of years.

H. David Hibbitt:

The Panel has been organized to address the topic "Future Needs and Directions in Nonlinear Analysis." This is a very general subject, on which there will be a wide spectrum of opinion. I would like to take the opportunity to present a rather subjective view of the present state of nonlinear numerical analysis, as it relates to current design practice, and draw some conclusions about directions in which I hope we will be moving. The perspective of this viewpoint is my interest in trying to apply nonlinear analysis to practical problems of significance in design.

First of all, I would like to say that I generally agree with the widely held opinion that nonlinear analysis has great potential for developing "better" designs over a wide range of applications. However, it has been my experience that this apparently well-recognized potential of nonlinear numerical methods is rarely realized. Indeed, I find that the only routine use of nonlinear analysis in design is in cases where it is legislated (such as high temperature design of nuclear power plant components), or where no practicable alternative exists, such as in the design of offshore piping installation processes. In cases where design analysts have alternatives, they rarely commit themselves to using the nonlinear analysis capabilities that are apparently available, in spite of the apparent payback that should accrue to adoption of more realistic (and hence nonlinear) numerical models. Why?

I think the basic answer to this question is the rather low confidence level assigned to such analysis results, rightly or wrongly, by the designers who must use them. An immediate problem is that the terminology is too general - codes and problems are categorized as linear or nonlinear, as though these are two boxes of equal size, rather than a box of infinitesimal dimensions and an unbounded domain. Unfortunately, we try to avoid more precise categorization of problems; our geometric modeling capabilities - usually the finite element method - are rather general, as are the basic variational principles we build on, so we would like to see the same generality in our algorithmic approach. Experience suggests this is too optimistic. Confidence comes from experience, and I believe each member of the Panel could, if asked about a given problem, give a useful estimate of the confidence he would have in providing results of significance in design. The estimates may differ according to the respondent's experience. But, in any case, the key question is: "Can we rely on obtaining useful results for a predictable time and money budget?" The important word here is "reliability" - this means not only obtaining an answer, but having a basis for estimating how well that answer models reality - knowing the error. Nonlinear problems are, for the most part, complex. They require a great deal of judgement in modeling, in obtaining a solution once the problem is posed mathematically, and in interpreting the results as a design decision.

We need to minimize the demands for such judgements on the part of the analyst, and give him the basis for making those decisions which cannot be automated. Nonlinear analysis will approach its potential as we shift the burden of analysis judgements from the user to the software. My own involvement is with code development for production use, so that such concepts are of great interest to me. Ideally, the code would make all of the fundamental decisions beyond those of basic modeling - in particular, spatial and temporal errors would be bounded, as required, automatically. Apparently, this is not yet mathematically possible, certainly not in the general case, but we can give the user considerable help. Temporal error control appears to be the less difficult problem. Techniques are available for static, dynamic, and first-order problems; for example, the methods of Powell (Ref. 15) for static cases, Sutherland (Ref. 16) for explicit integration of creep and relaxation problems, or Park (Ref. 17) for structural dynamics. These have been shown to control such errors satisfactorily for many useful cases, certainly in "smoothly" nonlinear problems. Spatial error control requires spatial error estimates, and we still lack these for practical purposes (Ref. 18). The best we can do is give guidance and tell the user when such guidance is valuable and when it is not. Rather than seeking a universally applicable measure, those of us working on the use of codes in design should be developing validation of available estimates.

The analyst will always have to translate the physical problem into a mathematical one, and interpret the mathematical solution. It is unfortunately true that few analysts have been lucky enough to obtain a solid background in nonlinear mechanics. There is much room here for the imaginative application of currently available codes in teaching, and it is gratifying to see this begin to happen. Mr. Swanson makes his code freely available for this purpose, and Professors Bathe and Cleary taught a nonlinear mechanics course, using ADINA, just this past year at M.I.T.

What I would advocate here is the introduction of practically oriented numerical mechanics courses in engineering curricula, exposing students to a broad spectrum of mechanics concepts through the use of existing software, in contrast to the more common approach of teaching how to develop software to address a few, most commonly linearized, problems. I should emphasize that what I am suggesting here is to take advantage of existing software to provide a wider exposure to aspects of mechanics which present too many formidable or insurmountable analysis obstacles, except when treated numerically. This is a great challenge to educators, but is, in my judgement, a necessary step if we are to progress.

Thus, I am reducing our needs to the well-known requirements of verification and qualification. We should identify areas where the use of nonlinear analysis has its greatest potential for design improvements: the Sandia-sponsored program in light water reactor design (Ref. 19) is doing just this for one industry, but there are other applications, such as manufacturing, with as much potential benefit. Experimental work must be used to develop an adequate database for validation. We should provide designers with well-defined guidelines specifying problem classes in which nonlinear analysis is recommended, and how that analysis should be done. We should make the production codes as "user friendly" as possible. Finally, we should emphasize the need to qualify the analyst through education.

Part II - Comments From Audience and Panelists' Response

Moderator (Robert E. Nickell) opens the floor for discussion and comments.

Comment from Audience:

I'd like to address my comment to John Swanson. You mentioned that getting sophisticated packages within our general software system is an argument for continuation of general purpose programs. I surmise that it's an argument against general purpose programs and that in the future general types of analyses will be conducted by a series of packages that do certain parts of the analysis. For instance, one package will develop the mesh, another package will develop element stiffness matrices; other packages will solve the eigenvalue problem; other packages will solve for the material state given an increment in displacement or decide how much plastic or creep strain is accumulated in the increment. Each of these packages can be developed by a researcher who specializes in that area as long as he has some standards as to how he could store his data in a database. His package can interact with the other packages in the system so that the entire system would not be all manufactured by one organization, but would be manufactured by specialists who know the most about each area of the nonlinear analysis being conducted. I'd like to solicit comments from any of the panelists on that.

Response from Panelists:

H. D. Hibbitt:

If I could just comment on that. It seems to me that's precisely a description of a modern general purpose code which is basically a database management system with certain modules that do various operations on the data.

J. A. Swanson:

In fact, I was going to suggest that the definition you were giving sounded much like a person planning to develop the general purpose code and wanting to make sure he had the whole setup modularized to the point where he could have ten employees and yet produce an assured quality package.

Comment from Audience:

Do you plan to have any certification standards for these programs because as a user we are often confronted with the problem? Program documentation is totally inadequate most of the time. The vendors imply that they have so many problems they can solve and then they work for two months trying to solve your problem. Then he says: "Well, that option doesn't work well anyway, let's wait until we get a new version that works." We want to use this program to design. I think we are lacking professionalism. Do you have any plans to set up a standard for certification procedure? You imply that everything can be

solved by this program yet it has so many deficiencies it can hardly fly.

Response from Panelists:

R. E. Nickell:

Let me comment first and just set the stage. First of all, certification is a third party process. I think you will all agree, we are talking here about first party, mainly the gentlemen that are here and second party, mainly the users, and certification would imply a third party get involved. However, if you gentlemen choose to tackle a third party certification you are welcome.

J. F. Gloudeman:

Those of you who have MSC/NASTRAN delivered to you know that with the delivery comes a collection of sample problems and test cases. Our Quality Assurance program consists of more than 400 problems. But all of us face much the same problems. For example, the NRC and other government agencies have certain standards for use on selected contracts. I know that John and I would both do whatever we could to get a universal good housekeeping seal of approval because we have to go through this in individual companies, individual installations, and different commercial data centers. One problem is that the standards are not always uniform. If you have ideas on how we can cope with this, please let us hear from you.

J. A. Swanson:

I'm for that. I think it is unreasonable to ask the program developers to set standards of quality for themselves or to define their own test problems. Of course, I define my own and get perfect results. I submit a verification manual which has 100 or more problems. I challenge anyone else to show how they would solve test problems and I'll take their problems and show that I can solve them. But all we have proven, therefore, is that we can solve 200 problems. Unfortunately, the problem that you would have is not one of the 200. I think there have been theoretical studies that say it would take so many millions or billions of problems to exercise every possible path. There is a Latin phrase that says: "Let the buyer beware." I would put it another way: "Let the buyer try a couple of problems with a very critical eye and prove us wrong." We say that the software works. We welcome anyone, and that is one of the outshoots of our university program. The students tend to be as critical if not more critical than some of the people in the commercial environment. We are looking for anyone out there who can say that this doesn't work because as far as we know it does work. We have tried some things with it; it looks right. There are many areas in which theoretical solutions can be used to evaluate our software, for example, fracture mechanics or large deflection plasticity. If you have some solutions, we would be happy to work with you. We need more input, and you need a very critical eye. Don't start a big problem without assuring yourself that you have reasonable confidence that it's going to work. We are not guaranteeing it, but we are not in the business to go out of business either. Presumably we have some degree of reliability; otherwise we would not be sitting here after ten years.

K. J. Bathe:

I'd like to add that the interaction between the user and developer of the software is of utmost importance when considering nonlinear analysis. When we think of nonlinear problems to be solved, it can take months, sometimes even years, to solve these problems. You cannot expect the developers to demonstrate to you that your problem is really solvable in the way you would like to solve it. I find that I get telephone calls frequently of people saying: "I have this problem; can I solve it?" I can only say, "You can do something with ADINA on that problem, but whether or not you really get down to the level of the answer that you would like to see is a completely different question." One may have to perform an extensive study to see whether or not you can really obtain the answers that you are looking for.

R. E. Nickell:

I will complete that a little. Have any of you answered yes or no to that? Have any of you been asked to be bonded or have you negotiated on release of liability with potential users? Is that a fair question to ask? Is that getting to be quite common?

Response from Audience: Raised hands and comments.

R. E. Nickell:

I think at least two of them responded affirmatively that they would welcome some kind of third party certification process if I understood them correctly.

J. F. Gloudeman:

I have investigated the possibility of having organizations like NRC do their testing and give us a uniform good housekeeping seal of approval or certification but they apparently prefer to do it on a case by case basis for some reason. That's why I am seeking the help of the community. We'd certainly welcome a certification program. We now have to put into our lease agreements certain protective clauses. One of the things that John Swanson mentioned, the publication of an error list, is an important part of that. We feel an obligation not only to tell our users what's good about MSC/NASTRAN, but once a month they get our newsletter that states what's wrong with it. We hope users are reading that newsletter. I think a good testing program should cover not only what works, but also which algorithms are weak. If somebody wants to start a certification club, I'll join right now.

Comment from Audience:

I have also been wrestling with computer programs for the last few years, chasing bugs of my own and chasing bugs of others, and in a paper that appears in the proceedings of this symposium, Dr. Noor has compared several programs. He omits the one David Hibbitt is associated with and gives a line count of statements in each of these programs. I tried to put these in terms of something that I can handle intellectually so I have estimated the weight of each

program, assuming about five pounds per box of cards. Professor Bathe's program, ADINA, comes out to about 200 pounds, Dr. Swanson's program is about 600 pounds, the ABAQUS code is about 400 pounds, and MSC/NASTRAN comes out to about 3/4 ton. The questions that I have are: "What techniques do you use to find the bugs in these enormous haystacks?" and "What techniques do you use to prevent these errors from getting into the codes?"

R. E. Nickell:

I know this is a weighty discussion. I have no idea.

K. J. Bathe:

Modularity of the codes, which we referred to earlier, is part of the answer. We have stand-alone subroutines that we can test very well. I take care of the overall database management, and then I make sure that everything that is added to the program is very well checked out by itself, that is, every finite element, every material model, and so on. I must say though that I would not like to take care of a 3/4 ton program. I am satisfied with 200 pounds and I am trying to get slimmer. I believe the SAP IV computer program is probably worth about 100 pounds; however, the capabilities of ADINA are much more than SAP IV. What I personally have seen over the last six years from the release of SAP IV to the release of ADINA is a slimming down. I am trying to make the ADINA code smaller while at the same time expanding its capabilities. If we were to keep the same capabilities that we now have in ADINA, and were to spend the next years on simply making it smaller, we certainly would be successful. We would come down from 200 pounds to 100 pounds, 60 pounds, and so on. The objective, however, is to try at the same time to expand the capabilities, and what we generally find is that the size goes up. I certainly think that one of the objectives of a code developer should be modularity, and at the same time, trying to keep the size of the code down to a minimum.

J. F. Gloudeman:

I think it's a well-phrased question. I wish I had a well-phrased answer. I'll tell you what we do at the MacNeal-Schwendler Corporation to help cope with carrying around all that weight. First of all, we have a fairly large community that's been actively using our code for over a decade. As a result, the number of different possible paths in which the programmers can make mistakes and the designers can misdesign things has been pretty well run through. That alone is not enough. The feature of modularity referred to earlier does help a lot. MSC/NASTRAN has 430,000 source statements, and it will continue to grow because we intend to add new capability as needs arise. When we add new capability, we put our quality assurance team together from the time we start initial development. We have a quality assurance organization that is separate from both the engineering organization, where the ideas are developed and the algorithms are put together, and the computer science department, where the programs are actually developed. We feel that we need that kind of independence, and our testing team makes sure that both the algorithms that are designed, and the code that is developed, are checked out before new versions of the program are released. I mentioned earlier the set of more than 400 test

cases that we go through. We run through those same test cases prior to each release to avoid regression errors, that is, to make sure that what did work, continues to work. We try to go about it systematically, but we are much more concerned about efficiency and reliability than being the first out with a new code. We can't afford to release new things in a big hurry. Consider nonlinear behavior in MSC/NASTRAN, for example. We started with geometric nonlinearity back in 1978. It received enough attention to cause us to find ways to make it better and faster and run more reliably. The next release with the material nonlinearity will essentially feature static plasticity, and then a year later, probably creep and nonlinear transient response. Thanks to the tremendous feedback from our user community and going about it very systematically and slowly, we minimize the risk. Approximately 200 errors are reported per year and we discover 98 percent of them before the system hits the street. Our user community, as far as we are concerned, is operating in a rather stable mode in that the number of error reports for MSC/NASTRAN is decreasing each year. We feel that is to our credit.

Comment from Audience:

My name is Sam Key from Sandia Laboratories. I have found graduate courses in theoretical continuum mechanics and tensor analysis barely adequate for writing, thinking, and developing nonlinear finite element codes. I think the problem resides with the people and not with software. I think it's an educational problem. I am really concerned about how we are going to train people adequately in using software intelligently. I would like to have the panel respond to that.

Response from Panelists:

H. D. Hibbitt:

It's very easy for me to respond to that. I agree entirely.

R. E. Nickell:

Yes, I think Dave commented at least twice on that question.

J. F. Gloudeman:

It's a major area of concern for us. We have expanded our own live teaching classes. In addition, the Schaeffer Analysis organization which is part of the MacNeal-Schwendler Corporation has now started developing video tape courses which we know some people want. We're also looking into computer-assisted education. My own personal view is that eyeball to eyeball teaching is the best way to go, and we continue to emphasize this. We will also make available video tape courses to those people who think that's the best way to train their organizations. Through these video tapes we can essentially duplicate our staff many fold and teach uniform quality courses on an international basis. Another way we are addressing the education challenge is by making MSC/NASTRAN available to universities at special rates. Universities that are now setting up CAD/CAM programs are teaching from two different approaches:

basic finite element modeling and further development of available finite element codes. The first deals with discretizing a structure, how you model. It's an art and it's a tough field. I don't know of any published textbooks on this subject. We hope that the universities will produce one or more while teaching people how to use existing codes effectively. The second approach is more difficult. Professor Bathe stated earlier that it takes three years for a student to really develop something halfway decent. What we are trying to do with MSC/NASTRAN is to provide universities with an existing code whose fundamental algorithms already work so that research can take place in such a way that the students don't have to start from scratch and develop a whole new code. For example, they can concentrate on new elements or new algorithms they want to test, and only add those parts to an otherwise already working code. The use of a higher-order language like DMAP facilitates such efforts.

K. J. Bathe:

With regard to the course that Dave Hibbitt already mentioned, we had a very good response. In this course, "Theory and Practice of Continuum Mechanics," classical continuum mechanics principles and the use of finite element methods operating on these principles were discussed. Then the students used ADINA to solve interesting problems, sometimes just single element problems of nonlinear elasticity. They printed stresses and strains and checked the quantities for large displacements and strains by hand, making sure everything was correct. I am very excited about the course, and I think more universities will offer similar courses. Video tape courses are also very attractive. I just completed a short course on video tape, prepared by the Center of Advanced Engineering Study, M.I.T. Further, I think there are big gaps in the textbook writing area. When we wanted to prepare our notes for the course at M.I.T., there was really no textbook available that we could use. So I started writing, and I just delivered to Prentice Hall a manuscript which I hope will be a contribution in the field of teaching engineers how to use finite element codes, how to understand what is in the code, and so on. But I agree fully that it is, to a very large extent, an educational problem.

REFERENCES

1. Oden, J. T., and Bathe, K. J., "A Commentary on Computational Mechanics," Applied Mechanics Reviews, Vol. 31, No. 8, August 1978.

2. Bathe, K. J., "On the Current State of Finite Element Methods and Our ADINA Endeavours," Adv. Eng. Software, Vol. 2, No. 2, 1980.

3. Course 2.094, Theory and Practice of Continuum Mechanics, Bulletin, Massachusetts Institute of Technology, 1979/80.

4. Corum, J. M., "Future Needs for Inelastic Analysis in Design of High-Temperature Nuclear Plant Components," in Computational Methods in Nonlinear Structural and Solid Mechanics, edited by Ahmed K. Noor and Harvey G. McComb, Jr., held October 6-8, 1980, Washington, D.C., pp. 231-240.

5. Mikulas, M. M., Jr., "Failure Prediction Techniques for Compression Loaded Composite Laminates With Holes," in NASA CP-2142, Selected NASA Research in Composite Materials and Structures, August 1980, pp. 1-33.

6. Noor, A. K., "Recent Advances in Reduction Methods for Nonlinear Problems," in Computational Methods in Nonlinear Structural and Solid Mechanics, edited by Ahmed K. Noor and Harvey G. McComb, Jr., held October 6-8, 1980, Washington, D.C., pp. 31-44.

7. Noor, A. K., "Survey of Computer Programs for Solution of Nonlinear Structural and Solid Mechanics Problems," in Computational Methods in Nonlinear Structural and Solid Mechanics, edited by Ahmed K. Noor and Harvey G. McComb, Jr., held October 6-8, 1980, Washington, D.C., pp. 425-465.

8. Lecht, Charles P., "Waves of Change," Second Edition, McGraw-Hill, New York, New York, 1979.

9. Hu, K. K., Swartz, S. E., and Huang, C. J., "A Proposed Generalized Constitutive Equation for Nonlinear Para-Isotropic Materials," NASA Conference Publication 2147, presented at a Symposium in Washington, D.C., October 6-8, 1980.

10. Zimmermann, T. K., and Liu, W. K., "Finite Elements for Contact Problems in Two-Dimensional Elastodynamics," NASA Conference Publication 2147, presented at a Symposium in Washington, D.C., October 6-8, 1980.

11. Hussain, N., Khozeimeh, K., and Toridis, T. G., "Inelastic Behavior of Structural Components," NASA Conference Publication 2147, presented at a Symposium in Washington, D.C., October 6-8, 1980.

12. Siu, D., and Turcke, D., "Optimal Finite Element Discretization for Nonlinear Constitutive Relations," in Computational Methods in Nonlinear Structural and Solid Mechanics, edited by Ahmed K. Noor and Harvey G. McComb, Jr., held October 6-8, 1980, Washington, D.C., pp. 83-87.

13. Larkin, L. A., "Elasto-Plastic Analysis With NASTRAN User Elements," in Computational Methods in Nonlinear Structural and Solid Mechanics," edited by Ahmed K. Noor and Harvey G. McComb, Jr., held October 6-8, 1980, Washington, D.C., pp. 357-362.

14. Lawson, D. W., Webster, J. L., and Overbye, V. D., "MSC/NASTRAN Superelements in Structures With Reflective Symmetry," MSC's 1980 Users Conference.

15. Powell, G., and Simons, J., "Improved Iteration Strategy for Nonlinear Structures," Department of Civil Engineering, University of California at Berkeley, April 1980.

16. Sutherland, W. H., and Watwood, V. B., "Creep Analysis of Statically Indeterminant Beams," Battelle N.W. Labs., Report 1362, June 1970.

17. Felippa, C. A., and Park, K. C., "Direct Time Integration Methods in Nonlinear Structural Mechanics," Computer Methods in Applied Mechanics and Engineering, Vol. 17/18, 1979, pp. 277-313.

18. Babuska, I., and Rheinboldt, W. C., "Reliable Error Estimation and Mesh Adaptation for the Finite Element Method," Institute for Physical Science and Technology, University of Maryland, April 1979.

19. Nonlinear Analysis in Light Water Reactor Design, reports to the Light Water Reactor Safety Technology Management Center, Sandia Laboratories, 1980.

BASIC DISCIPLINES IN THE DEVELOPMENT
OF ANALYSIS PROCEDURES

A. APPROXIMATION THEORY

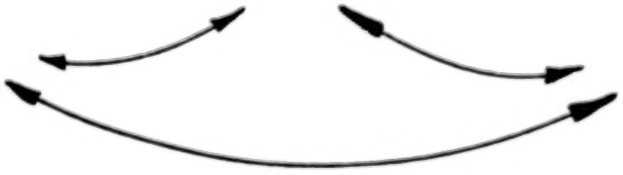
1. FORMULATION OF NONLINEAR CONTINUUM MECHANICS EQUATIONS OF MOTION
2. DEVELOPMENT OF FINITE ELEMENT EQUATIONS
3. IDENTIFICATION AND DEVELOPMENT OF MATERIAL MODELS

B. NUMERICAL METHODS

1. NUMERICAL INTEGRATION IN SPACE
2. TIME INTEGRATION
3. SOLUTION OF EQUATIONS
4. CALCULATION OF EIGENSYSTEMS

C. COMPUTER TECHNIQUES

1. PROGRAMMING METHODS
2. USAGE OF AVAILABLE HARDWARE AND SOFTWARE
3. EFFICIENT PROGRAM ORGANIZATION
4. FLEXIBILITY FOR MODIFICATIONS



INTERACTION

Figure 1

PREREQUISITES FOR ADVANCING THE STATE-OF-THE-ART IN NONLINEAR ANALYSIS

TO ADVANCE THE STATE OF NONLINEAR ANALYSIS PROCEDURES WE NEED EMPHASIS ON: BASIC, GENERAL AND RELIABLE METHODS.

- Continuum Mechanics and Finite Element Formulations
- Constitutive Models
- Numerical Methods
- Computer Program Implementations

Figure 2

POSSIBLE PITFALLS IN SOLUTION
OF NONLINEAR PROBLEMS

REMEMBER:

{Inappropriate Theoretical Assumptions}

Plus

{Program Bugs}

May Give

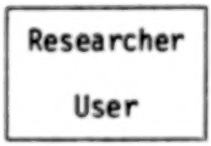
{Good Correlation with Experimental Results}

Figure 3

RESEARCHER VERSUS USER OF A PROGRAM

DISTINGUISH BETWEEN:

- Developing a procedure for analysis of a very specific problem.



- Designing a method that is generally applicable to a whole problem area.

Figure 4

OUTLINE OF J. GLOUDEMAN PRESENTATION

- Primary Forcing Function; Computer Hardware
- Changing User Requirements
- Challenges to Systems Developers

Figure 5

PRIMARY FORCING FUNCTION; COMPUTER HARDWARE

- CPU
  - 1980: 10 MFLOPS
  - 1985: JOSEPHSON DEVICES
    - 100 TIMES FASTER PER LECHT
    - 10,000 TIMES FASTER PER FUBINI
    - $30 PER MEGABIT STORAGE CHIP (FUBINI)
- IMPACT OF SUPERMINICOMPUTERS
- WORD SIZE (NOMINALLY 64 BITS)
- 1980 - UP TO 16 MEGA WORDS

  1985 - 100 MEGA WORDS
- PERIPHERAL STORAGE (DIRECT ACCESS)
  - 1980 500 MB DEVICES:
    UP TO 100 DEVICES
  - 1981 : IBM 3380 (2.52 BILLION CHARACTERS)
  - 1985 NON-ROTATING (?) TERABIT MEMORY
- DATA MOVEMENT
  - 1980 HYPERCHANNEL, 50 MBS
  - 1985 (?)

Figure 6

PRIMARY FORCING FUNCTION; COMPUTER HARDWARE (CONCLUDED)

- PRICE PERFORMANCE IMPROVEMENTS
  - CPU
  - MEMORY
  - PERIPHERAL STORAGE
- SOFTWARE LAGS; HELP FROM FIRMWARE (?)
- DATABASE MANAGEMENT SYSTEMS
  - TECHNICAL NEEDS DIFFERENT FROM BUSINESS NEEDS
- STANDARDS
  - GEOMETRY, GRAPHICS, TELECOMMUNICATIONS

Figure 6 (Concluded)

JOSEPHSON TUNNEL JUNCTION (1962)

- SUPERCONDUCTORS (LOGIC AND MEMORY)
  - OPERATE AT TEMP OF LIQUID HELIUM (4.2°K)
- SWITCHING SPEEDS IN PICOSECOND RANGE
  - 100 TO 200 TIMES FASTER THAN TODAY'S SEMICONDUCTORS
- THREE-LAYER, LAMINATED CONSTRUCTION
  - OUTER LAYERS OF SUPERCONDUCTING MATERIAL
  - SEPARATED BY AN INSULATING LAYER
- GENERATE 1/1000 HEAT OF CURRENT TECHNIQUES
- PACKAGED IN LIQUID HELIUM
  - OUTER CHAMBER OF LIQUID NITROGEN

Figure 7

MAJOR BREAKTHROUGHS NEEDED

- DATABASE MANAGEMENT SYSTEMS
- GEOMETRIC REPRESENTATION (STANDARDS)

Figure 8

CHANGING USER REQUIREMENTS

- CRASH DYNAMICS (NONLINEAR TRANSIENT RESPONSE)
  - AUTOMOTIVE
  - HELICOPTERS
- NUCLEAR POWER INDUSTRY
  - NONLINEAR MATERIALS
  - TEMPERATURE, PRESSURE EXTREMES
  - SEISMIC ANALYSIS
- FATIGUE AND FRACTURE MECHANICS
- MILITARY: ARMORED VEHICLES VERSUS LASERS
- HEAVY INDUSTRY: EARTH MOVERS
- OFFSHORE PLATFORMS
- SHIP STRUCTURES (LNG TANKERS)
- ENERGY DISTRIBUTION
  - PIPING ANALYSIS

Figure 9

CHALLENGES TO SYSTEMS DEVELOPERS

- PHYSICAL BEHAVIOR MODES REASONABLY WELL UNDERSTOOD
- VALIDITY OF ALGORITHMS
  - MATERIAL PROPERTIES
  - TIME/TEMPERATURE DEPENDENCIES
  - "MEMORY" OF MATERIAL
  - LEGITIMACY OF MODAL REPRESENTATION
- DESIGN OPTIMIZATION

Figure 10

REALISTIC UPPER LIMITS

- PLASTICITY
  - SUBSTRUCTURING LARGELY ELIMINATES LIMITS FOR STATIC ANALYSIS
  - SUPERMINI'S HANDLE UP TO 3000 DOF
- NONLINEAR TRANSIENT RESPONSE
  - ARMEN: CRASH ANALYSIS UP TO 3000 DOF
- PROBABILISTIC (E.G., MATERIAL PROPERTIES)
  - SAME ORDER AS TRANSIENT ANALYSIS

Figure 11

Figure 11

WHERE TO PUT THE EMPHASIS?

- NUMERICAL ANALYSIS
  - CHALLENGE OF NEW FORMULATION
  - DRUDGERY OF REFINING TRADITIONAL ALGORITHMS
- DATABASE MANAGEMENT
- STANDARDS FOR GEOMETRY

Figure 12

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
Supplement to NASA CP-2147		

4. Title and Subtitle	5. Report Date
Research in Nonlinear Structural and Solid Mechanics	June 1981
	6. Performing Organization Code: 506-53-53-03
7. Author(s): Harvey G. McComb, Jr., and Ahmed K. Noor, compilers	8. Performing Organization Report No.: L-13950
	10. Work Unit No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Conference Publication Supplement
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A symposium on Computational Methods in Nonlinear Structural and Solid Mechanics was held in Washington, D.C., on October 6-8, 1980. NASA Langley Research Center and The George Washington University sponsored the symposium in cooperation with the National Science Foundation, the American Society of Civil Engineers, and the American Society of Mechanical Engineers. The purpose of the symposium was to provide a forum for communicating recent and projected advances in applied mechanics, numerical analysis, computer hardware and engineering software, and their impact on modeling and solution techniques in nonlinear structural and solid mechanics. The fields covered by the symposium are rapidly changing and are strongly impacted by current and projected advances in computer hardware. To foster effective development of the technology, therefore, an expert in computer hardware was invited to communicate in a keynote address his perceptions on new computing systems and their impact on computation. Also, five experts in nonlinear analysis software systems were invited to participate in a Panel Discussion on "Future Needs and Directions of Nonlinear Analysis." This NASA publication includes the keynote address, introductory remarks presented by the panelists, selected comments from the audience, and the responses by the panelists.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Nonlinear mechanics Structural mechanics Solid mechanics Computational methods	Unclassified - Unlimited Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	44	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1981

END

3-10-82